Laboratoire ARTEMIS
BP 53x - 38041 - Grenoble cedex

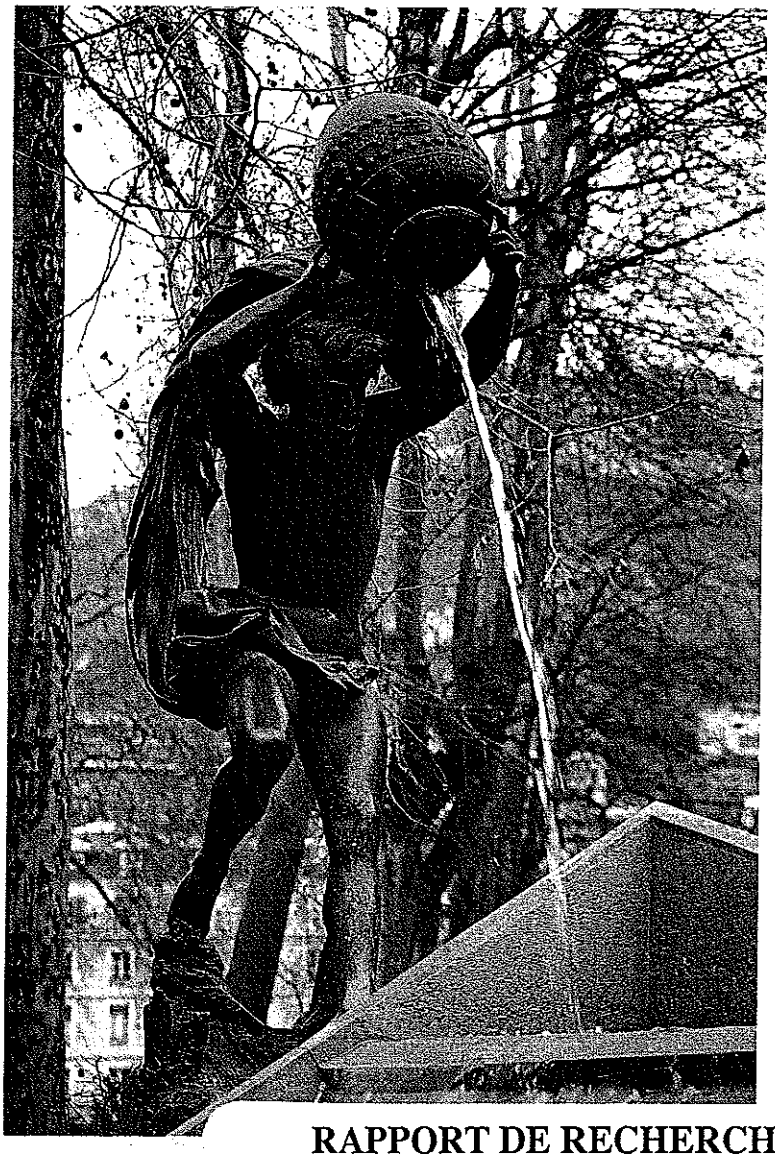

Photo Focale Studio Grenoble

RAPPORT DE RECHERCHE

Computational results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem

P. Augerat, J.M. Belenguer, E. Benavent

A. Corberan, D. Naddef, G. Rinaldi

RR 949 - M Septembre 1995

Centre national de la recherche scientifique
Ecole normale supérieure de Lyon

Institut IMAG

Institut national polytechnique de Grenoble
Université Joseph Fourier Grenoble I

Computational results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem

P. Augerat[1], J.M. Belenguer[2], E. Benavent[2],
A. Corberán[2], D. Naddef[1], G. Rinaldi[3]

1. ARTEMIS-IMAG. Grenoble France
2. Universitat de València. Spain
3. IASI-CNR. Rome

Abstract

In this paper, we present a Branch and Cut algorithm to solve the CVRP which is based in the partial polyhedral description of the corresponding polytope. The valid inequalities used in our method can be found in Cornuejols and Harche (1993), Harche and Rinaldi (1991) and in Augerat and Pochet (1995). We concentrated mainly on the design of separation procedures for several classes of valid inequalities.

The capacity constraints (generalized subtour eliminations inequalities) happen to play a crucial role in the development of a cutting plane algorithm for the CVRP. A large number of separation heuristics have been implemented and compared for these inequalities. There has been also implemented heuristic separation algorithms for other classes of valid inequalities that also lead to significant improvements: comb and extended comb inequalities, generalized capacity inequalities and hypotour inequalities.

The resulting cutting plane algorithm has been applied to a set of instances taken from the literature and the lower bounds obtained are better than the ones previously known. Some branching strategies have been implemented to develop a Branch an Cut algorithm that has been able to solve large CVRP instances, some of them which had never been solved before.

Keywords: Capacitated Vehicle Routing Problem, Branch and Cut, valid inequalities.

Acknowledgments: This work was partially financed by the EEC Science Program SC1-CT91-620 and by the project CE92-0007 of the Spanish Ministerio de Educación y Ciencia.

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) we consider in this paper consists in the optimization of the distribution of goods from a single depot to a given set of customers with known demand using a given number of vehicles of fixed capacity. There are many practical routing applications in the public sector such as school bus routing, pick up and mail delivery, and in the private sector such as the dispatching of delivery trucks. The reader can refer to Bodin et al. [BGAB83], Christofides [Chr85], Magnanti [Mag81] and Osman [Osm93b] for a survey of vehicle routing applications, model extensions and solution methods.

In the model we deal with in this paper, there are n customers indexed by i, each one with a demand of d_i units of goods to be delivered from the depot, and k vehicles of capacity C. Each customer i must be assigned to exactly one vehicle, i.e. demand splitting is not allowed and the demand d_i must be entirely delivered by one vehicle. The total demand assigned to each vehicle must satisfy the capacity constraint, i.e. can not exceed C. The objective is to minimize the total distance traveled by the vehicles. So, apart from assigning customers to vehicles, one has to determine for each vehicle the minimum length route leaving the depot, visiting all customers assigned to it and going back to the depot. In this paper, we restrict our attention to symmetric distance functions in the objective.

If the capacity C is large enough (at least $\sum_{i=1}^{n} d_i$), then it is possible to assign all customers to a single vehicle and the capacity constraints are redundant. In this case, the CVRP reduces to the multi-salesman traveling salesman problem (k-TSP). On the other hand, if the capacity C is small according to the demands, the capacity limitation becomes a crucial part of the problem and one has to solve a bin packing problem for the assignment of customers (or items of weight d_i) to vehicles (or bins of size C). Since k-TSP and bin packing are both NP-hard problems, CVRP looks like the intersection of two difficult problems and is also an NP-hard problem. This is the reason why routing problems have first been tackled using heuristic approaches. See again the above mentioned references and for recent results Hiquebrau et al. [HASG94], Taillard [Tai93], Osman [Osm93a], Gendreau et al. [GHL94] and Campos and Mota [CM95].

During the past fifteen years, exact algorithms have also been developed to solve capacitated routing problems of reasonable size to optimality. For example, Agarwal et al. [AMS89], Mingozzi et al. [MCH94] use a set partitioning and column generation approach and Fisher [Fis94] uses a lagrangian approach based on the minimum k-tree relaxation. Other exact approaches are presented in the surveys of Christofides [Chr85] and Laporte [Lap92].

Another approach to solve CVRP to optimality is the polyhedral approach which has proved to be efficient to solve large instances for the TSP. For example, Padberg and Rinaldi [PR91] report on the resolution of problems with up to 2392 cities.

Note that, "large instance" does not have the same meaning for VRP and for TSP. What we call today a large instance for CVRP is a problem with 100 customers. Fisher [Fis94]

reports on the solution of some test problems with up to 100 customers to optimality using a lagrangian relaxation approach embedded in branch and bound. On the other hand, some standard test problems with 76 customers have never been solved to optimality.

Initial investigations in the polyhedral aspects of the identical customers CVRP and in the similarities between the TSP and CVRP polyhedra were performed by Araque [Ara90], Campos et al. [CCM91], Araque et al. [AHM90], Laporte and Nobert [LN84] and Laporte et al. [LND85]. A more complete description of the CVRP polyhedron can be found in Cornuejols and Harche [CH93] and in Augerat and Pochet [AP95]. Computational results using the polyhedral approach are reported in Araque et al. [AKMP94] (for the identical customer case), Cornuejols and Harche [CH93] and Laporte et al. [LND85]. They solved moderate size problems with up to 60 customers.

The content of the paper is as follows. In Section 2, we describe formally the CVRP and the valid inequalities that we use in our "Branch and Cut" code. In Section 3, we present separation procedures for the capacity constraints (generalized subtour elimination inequalities). In Section 4, we show that other inequalities lead to significant improvements in some kind of instances. Section 5 is dedicated to branching strategies and the paper concludes in Section 6 with some computational results for a set of difficult instances taken from the literature.

2 Problem Description and Known Classes of Valid Inequalities

We present in this section the basic CVRP formulation and notation we will be using in the sequel. We then present briefly most of the known classes of valid inequalities for this formulation. The formulation we use is the so-called vehicle flow formulation with two indices. Its main advantage resides in the small number of variables needed. By removing the depot from the formulation, one could obtain an equivalent formulation of the path partitioning type (see Araque et al. [AHM90]).

Problem Description

Let $G(V, E)$ be an undirected and complete graph with node set V containing $n+1$ nodes numbered $0, 1, \ldots, n$. The distinguished node 0 corresponds to the depot and the other nodes correspond to the n customers. The set of customers is denoted by V_0, so $V = V_0 \cup \{0\}$. For each customer $i \in V_0$, we are given a positive demand d_i. For each edge $e \in E$, we are given a positive cost value b_e which corresponds to the distance between the two end-nodes of e (cost of traveling along edge e). Let k be the fixed number of vehicles available at the depot and C the constant vehicle capacity.

For a given subset F of edges of E, $G(F)$ denotes the subgraph $(V(F), F)$ induced by F where $V(F)$ is the set of nodes incident to at least one edge of F.

A **route** is defined as a nonempty subset F of edges of E for which the induced subgraph $G(F)$ is a simple cycle containing the depot 0 (i.e. $0 \in V(F)$, $G(F)$ is connected and the degree of each node of $V(F)$ in $G(F)$ is 2) and such that the total demand of nodes (customers) in $V_0(F) = V(F) \setminus \{0\}$ does not exceed the vehicle capacity C. Such a route represents the trip of one vehicle leaving the depot, delivering the demand of nodes in $V_0(F)$ (traveling along the edges F) and going back to the depot. The length of a route is the sum of the traveling costs b_e over all edges $e \in F$. Note that if $|V(F)| = 2$ in a route F, then the route is composed by twice the same edge. In all other cases (i.e. $|V(F)| > 2$), an edge can appear only once in a route.

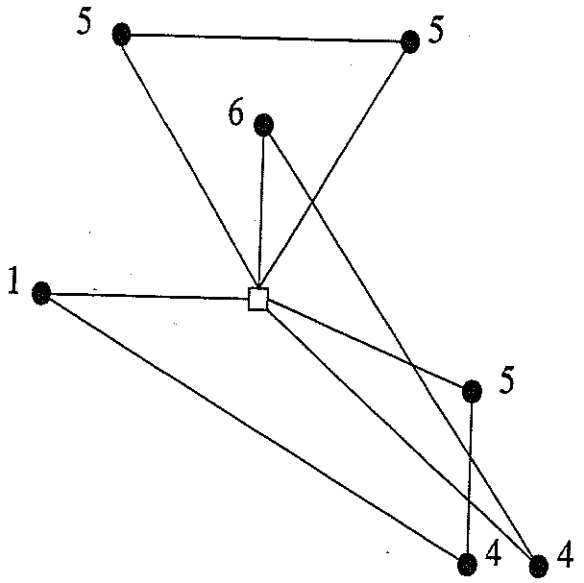

Figure 1: Instance of CVRP with $n = 7$ customers, $k = 3$ vehicles of capacity 10

A **k-route** is defined as a subset R of edges of E that can be partitioned into k routes $R_1, R_2, \ldots, R_k$ and such that each node $i \in V_0$ belongs to exactly one of the k routes (i.e. to exactly one of the sets $V(R_j)$, $1 \leq j \leq k$). The length of a k-route is the sum of the lengths of the k different routes defining it. Each k-route defines a feasible solution to CVRP and

the optimization problem consists in finding a minimum length k-route. Figure 1 represents the graph associated to an instance of CVRP with 7 customers and 3 vehicles, as well as a feasible 3-route.

Notation

In order to formulate the CVRP as an integer program, we associate a variable x_e to each edge e of E which will represent the number of times edge e is used by the solution (k-route). Sometimes, we will write x_{ij} instead of x_e where e is the edge between nodes i and j, $i < j$.

For simplicity, we will also use the following notations. For given subsets of nodes $S, S' \subseteq V$, the set $\delta(S)$ (coboundary of S) is the set of edges with one end-node in S and the other in $V \setminus S$ ($\delta(S) = \{e = (i,j) \in E : i \in S,\ j \in V \setminus S\}$), the set $(S : S')$ (cut between S and S') is the set of edges with one end-node in S and the other in S' ($(S : S') = \{e = (i,j) \in E : i \in S,\ j \in S'\}$), $\gamma(S)$ is the set of edges with both end-nodes in S ($\gamma(S) = \{e = (i,j) \in E : i,j \in S\}$) and $d(S)$ is the total demand in the set S ($d(S) = \sum_{i \in S} d_i$). For any subset F of edges, $x(F)$ denotes the sum of the x_e values over all edges $e \in F$.

CVRP Formulation

Now we can formulate the CVRP as the following integer program:

$$
\begin{aligned}
\text{(CVRP)} \qquad & \min \sum_{e \in E} b_e\, x_e && && (2.1)\\
& x(\delta(\{0\})) = 2k && && (2.2)\\
& x(\delta(\{i\})) = 2 && \textit{for all } i \in V_0 && (2.3)\\
& x(\delta(S)) \geq 2 \left\lceil \frac{d(S)}{C} \right\rceil && \textit{for all } S \subseteq V_0,\ S \neq \emptyset && (2.4)\\
& 0 \leq x_e \leq 1 && \textit{for all } e \in \gamma(V_0) && (2.5)\\
& 0 \leq x_e \leq 2 && \textit{for all } e \in \delta(\{0\}) && (2.6)\\
& x_e \textit{ integer} && \textit{for all } e \in E && (2.7)
\end{aligned}
$$

Constraint (2.2) states that each of the k vehicles has to leave and go back to the depot. Constraints (2.3) are the degree constraints at each customer node. Constraints (2.4) are the capacity constraints (also called generalized subtour elimination constraints), where $\lceil \alpha \rceil$ is the smallest integer larger or equal to α. They express that for a given subset S of customers, at least $\lceil d(S)/C \rceil$ vehicles are needed to satisfy the demand in S and, because the depot is outside S, each of these vehicles must necessarily enter and leave S. Furthermore, like for the subtour elimination constraints in the TSP case, these constraints are needed to force the connectivity of the solution. Constraints (2.5) – (2.7) specify the integrality and bound restrictions on the variables. A variable x_e linking a customer node directly to the depot (in (2.6)) can take the value 2 when the edge e is the only one in a route. The set of feasible solutions to CVRP (i.e. solutions to (2.2) – (2.7)) is called X^{CVRP}.

Based on this formulation, we have implemented a linear programming (LP) based cutting plane algorithm which proceeds in the following way. At each iteration, a linear program including the degree constraints and some set of valid inequalities is solved. If the optimal LP solution corresponds to a k-route, we are done. Otherwise, we strengthen the linear program by adding a set of valid inequalities violated by this optimal solution, and proceed as before. This cutting plane algorithm has been integrated in a Branch and Cut scheme to solve the CVRP.

We present now briefly the classes of valid inequalities for X^{CVRP} that we use in our cutting plane algorithm and in the next section, we will present the identification procedures we have implemented to find valid inequalities that are violated by a given LP solution.

Capacity Constraints

Denote by $r(S)$ the minimum number of vehicles needed to satisfy the customers demand in a set S in any feasible solution. We obtain the following valid inequality

$$x(\delta(S)) \geq 2\, r(S) \quad for\ all\ S \subseteq V_0,\ S \neq \emptyset \tag{2.8}$$

It is NP-hard to compute $r(S)$ but the inequality remains valid if $r(S)$ is replaced by any lower bound of $r(S)$. Cornuejols and Harche describe several lower bounds in [CH93]. We chose to use the inequality (2.4) instead of (2.8) because computing $r(S)$ appeared to be costly and useless for almost all the instances we tested.

Generalized capacity inequalities

Let $\Omega = \{S_i, i \in I\}$, be a partition of V_0 and let $I' \subseteq I$ denote the set of indices of those subsets S_i with more than one element (these subsets are also called clusters or supernodes). We define $r(\Omega)$ as the solution of the bin packing problem where the capacity of the bins is C and the set of items and their weights are defined as follows; for each $S_i, i \in I$:

- If $d(S_i) \leq C$, then include an item of weight $d(S_i)$,
- Otherwise, include $\lceil \frac{d(S_i)}{C} \rceil - 1$ items of weight C and one more item of weight $d(S_i) - (\lceil \frac{d(S_i)}{C} \rceil - 1)C$.

Then, it can be shown that the following inequality is valid:

$$\sum_{i \in I'} x(\delta(S_i)) \geq 2 \sum_{i \in I'} \lceil \frac{d(S_i)}{C} \rceil + 2(r(\Omega) - k) \tag{2.9}$$

This is a particular case of the generalized capacity constraints of Harche and Rinaldi [HR91]. Note that if $r(\Omega) \leq k$, then (2.9) is dominated by the capacity constraints. Otherwise, the validity of (2.9) can be informally explained as follows: let us assume for simplicity that $d(S_i) \leq C$, for all $i \in I'$ and that $r(\Omega) = k + 1$, then at least one set of customers, say S_j, will be visited by at least two vehicles, so $x(\delta(S_j)) \geq 4$.

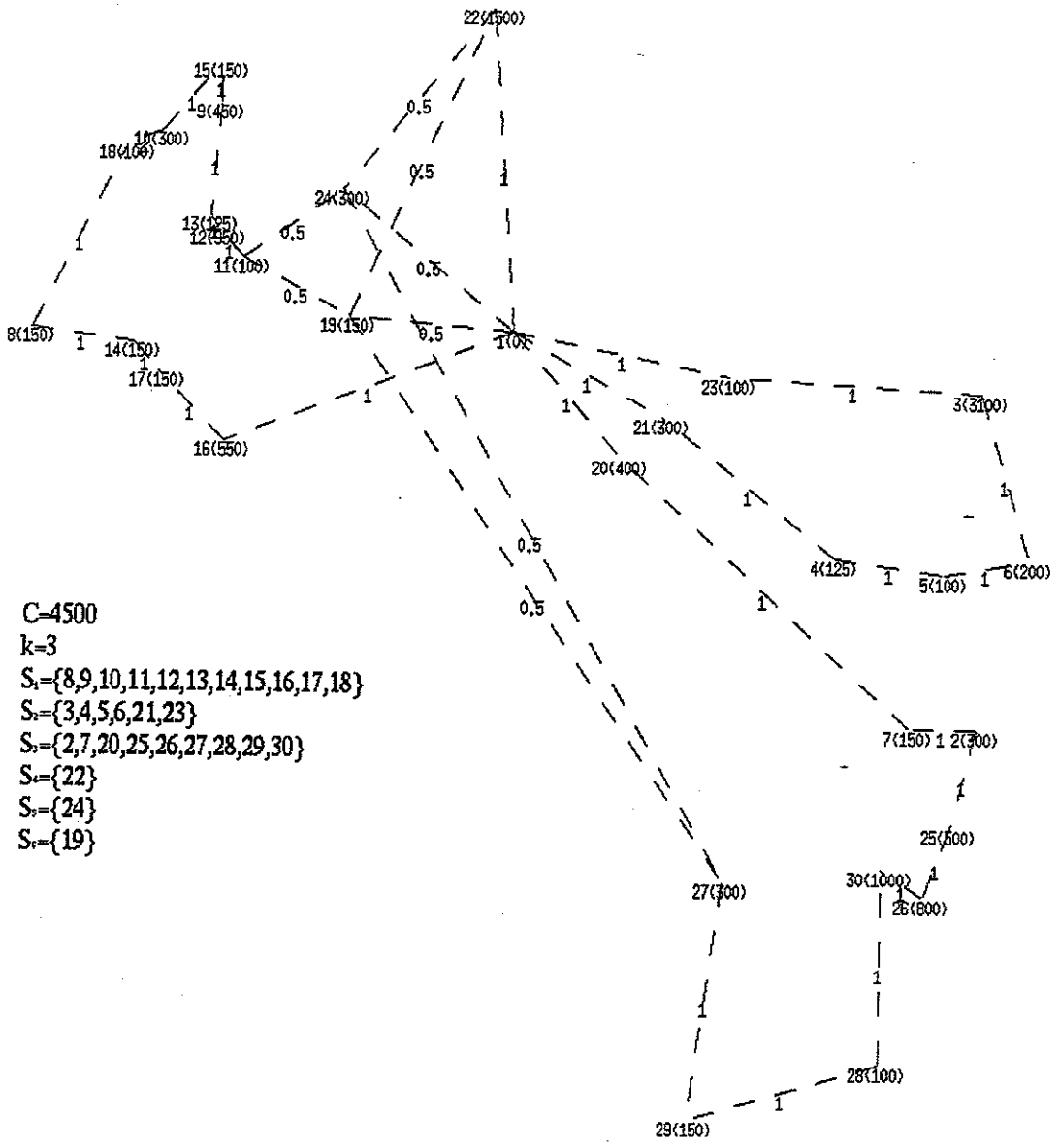

Figure 2: Violated generalized capacity inequality in E-n30-k3

Figure 2 corresponds to the support graph associated to a given fractional solution. The depot is here represented by node one and the demands are the numbers between parenthesis. Sets $S_i,\ i = 1,\ldots,6$ all have their coboundary equal to 2. Yet the bin packing problem with bins having capacity 4500 and 6 items having weights $d(S_1) = 3175, d(S_2) = 3925, d(S_3) = 3700, d(S_4) = 1500, d(S_5) = 300$ and $d(S_6) = 150$, gives a value of 4 while only 3 vehicles are available. The inequality $\sum_{i=1}^{3}\ x(\delta(S_i))\ \geq 8$ is violated.

When the same idea is applied on a subset H of V_0, we have a special case of the path-bin inequality introduced by Augerat and Pochet [AP95]. Let $H \subset V_0$ and $\Omega = \{S_i, i \in I\}$ be a partition of H and let I' be defined as above. If $r(H,\Omega)$ is the solution of the bin-packing problem on Ω, we have the bin inequality

$$x(\delta(H))\ +\ \sum_{i\in I'} x(\delta(S_i)) \geq\ \ 2\ \sum_{i\in I'} \lceil \frac{d(S_i)}{C} \rceil\ +\ 2\ r(H,\Omega) \tag{2.10}$$

Comb inequalities

Let $T_i,\ i = 1,\ldots,s$ and H be subsets of V_0 satisfying :
(i) $T_i \setminus H \neq \emptyset,\ i = 1,\ldots,s$
(ii) $T_i \cap H \neq \emptyset,\ i = 1,\ldots,s$
(iii) $T_i \cap T_j = \emptyset,\ i = 1,\ldots,s,\ j = 1,\ldots,s,\ i \neq j$
(iv) s odd

Then, the comb inequality (see Grötschel & Padberg [GP79]):

$$x(\gamma(H)) + \sum_{i=1,\ldots,s} x(\gamma(T_i)) \leq |H| + \sum_{i=1,\ldots,s} (|T_i| - 1) - (s+1)/2 \tag{2.11}$$

was proved to be valid for the CVRP by Laporte & Nobert [LN84]. Using (2.2) and (2.3), it is easy to see that (2.11) is equivalent to:

$$x(\delta(H)) + \sum_{i=1,\ldots,s} x(\delta(T_i)) \geq 3\ s\ +\ 1 \tag{2.12}$$

We consider now the case where the depot is "inside" the comb. Inequalities specific to the CVRP can be obtained considering special teeth and handles. Let $H,\ T_i,\ i \in I$ be subsets of $V_0 \cup \{0\}$ satisfying :
(i) $T_i \setminus H \neq \emptyset,\ i = 1,\ldots,s$
(ii) $T_i \cap H \neq \emptyset,\ i = 1,\ldots,s$
(iii) If $T_i \cap T_j \neq \emptyset$, for some $i, j, i \neq j$, then $0 \in T_i \cap T_j$ and
either $T_i \cap T_j \cap H = \emptyset$ or $(T_i \cap T_j) \setminus H = \emptyset$
(iv) s is odd

Let us define $c(T_i) = r(V \setminus T_i)$, if $0 \in T_i$ and $c(T_i) = r(T_i \setminus H)$ otherwise. Then, the extended comb inequality is:

$$x(\delta(H)) + \sum_{i=1,\ldots,s} x(\delta(T_i)) \geq 2 \sum_{i=1,\ldots,s} c(T_i) +\ s\ +\ 1 \tag{2.13}$$

Figure 3 represents another support graph of a fractional solution. The depot is node 1 and the demands are in between parenthesis. A comb is represented by solid lines: the handle H with bold edges and the teeth T_1, T_2, T_3 with thin edges. The corresponding comb inequality is violated since $x(\delta(H)) + x(\delta(T_1)) + x(\delta(T_2) + x(\delta(T_3)) = 11.33 < 2\,(r(T_1 \setminus H) + r(V_0 \setminus T_2) + r(T_3 \setminus H)) + s + 1 = 2\,(1+2+1) + 3 + 1 = 12$.

Hypotour inequalities

This class of inequalities are in the spirit of the hypo-hamiltonian inequalities for TSP introduced by Grötschel and Padberg [GP85]. Let H be a set of edges and $v \in V_0$. If H intersects all the routes including v then, the following inequality is a valid hypotour inequality:

$$x(H) \geq 1 \tag{2.14}$$

More general hypotour inequalities can be defined as follows. Let $\alpha x \geq \alpha_0$ be a valid supporting inequality such that αx is even for every feasible solution. If H intersects the set of edges of any feasible solution which satisfies $\alpha x = \alpha_0$, then the following is a valid extended hypotour inequality.

$$\alpha x + 2x(H) \geq \alpha_0 + 2 \tag{2.15}$$

We will concentrate in section 4.3 on identifying some classes of inequalities of this type. See Augerat and Pochet [AP95] for a complete description of them.

3 Identification of capacity constraints

Let x be the optimal solution of the LP at any iteration and let G(x) be the graph induced by the edges e with value $x_e > 0$. This graph is called the support graph of x.

Harche and Rinaldi [HR91] showed that the separation problem for the capacity constraints (2.4) is NP-Complete and designed several heuristics that can be briefly described as follows:

1) Check (2.4) for the node set of each connected component of G(x) and $G(x) \setminus \{0\}$.

2) Recursively, shrink edges with value $x_e \geq 1$ and non incident with the depot (see Padberg and Rinaldi [PR91] for a detailed description of the procedure in the TSP context). Shrinking an edge means to substitute the two endpoints of e by a supernode with demand equal to the sum of the demands of its endpoints. If the supernode has a demand greater than C, the set of vertices in the supernode violates (2.4).

   Let GS(x) be the shrunk graph thus obtained. It can be easily shown that finding a violated capacity constraint on G(x) is equivalent to finding it on GS(x). So, from now on, the heuristics in this section will be executed on GS(x), but for notational convenience we will assume that they are applied on G(x).

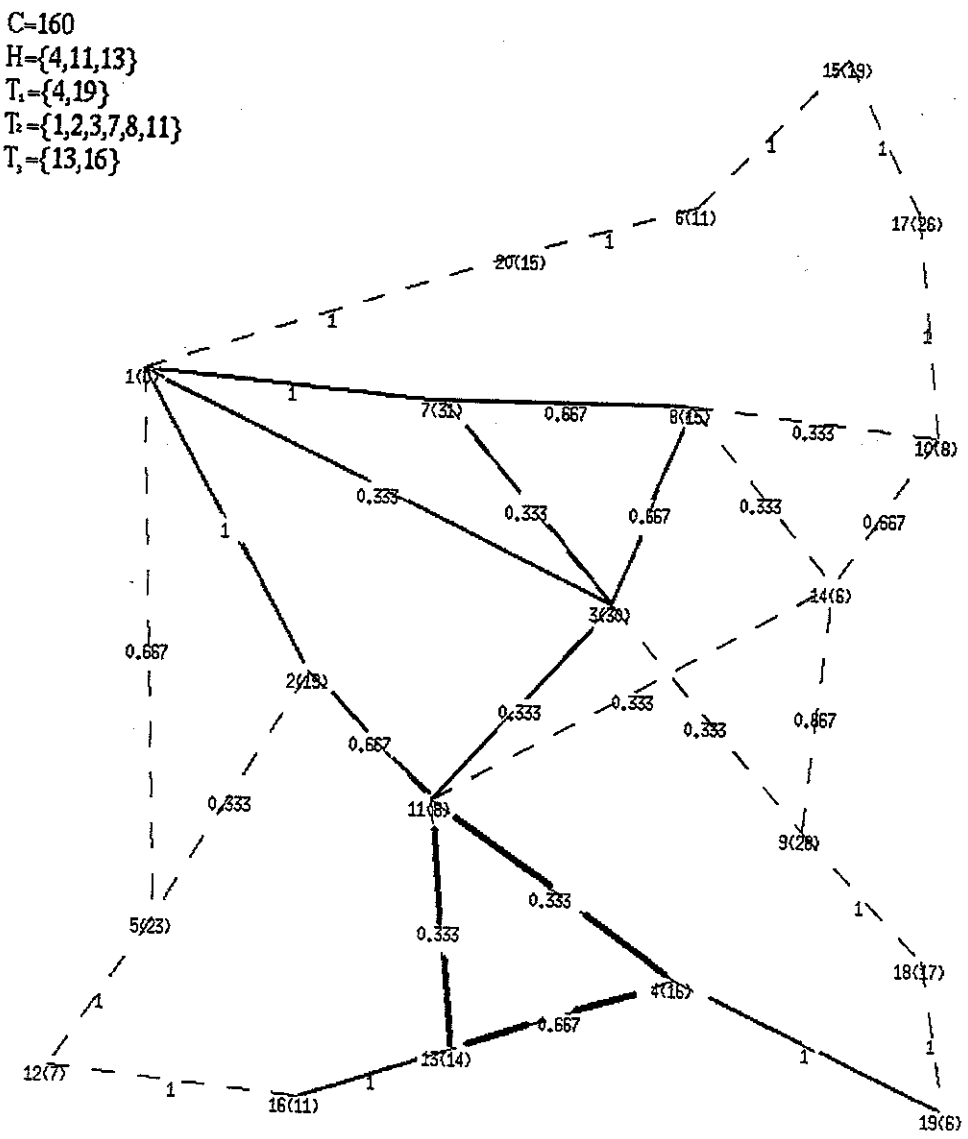

Figure 3: Violated comb inequality

3) Fractional capacity inequalities are obtained by replacing $\lceil \frac{d(S)}{C} \rceil$ in (2.4) by $\frac{d(S)}{C}$. Let $f(S) = x(\delta(S)) - 2\frac{d(S)}{C}$. It is possible to find the set $S \subseteq V_0$ for which f(S) is minimum by using a max-flow algorithm. If $f(S) < 0$, the fractional capacity inequality and the corresponding (2.4) constraint are violated. Anyway, the set S is a candidate set for checking violation in (2.4).

We have implemented all of these procedures together with some easy ideas that may be used with any other procedure. Once set S has been checked in (2.4), it is an easy task doing the same for set $V_0 \setminus S$. Furthermore, sets with demand exceeding a little bit an integer multiple of C, are good candidates to check (2.4). Then, every time a set S for which $pC \geq d(S) \geq (p-\epsilon)C$, with p integer and $\epsilon = 0.33$, is found, then (2.4) is checked for all the sets $S \cup \{v\}$ where v is adjacent to at least one node of S. Again, it can be done very fast.

The results obtained with the Harche and Rinaldi procedures, including the above ideas, are presented in Table 1.

Greedy shrinking algorithm

In this procedure, given an initial set of nodes S, at each iteration, a node v is added to S in such a way that $x((S:v))$ is maximum (in order to minimize $x(\delta(S))$).

Possible candidates for being the initial node set S are:
1 - only one node
2 - the end-points of an edge
3 - a set corresponding to a tight inequality in the linear program
4 - the complementary of such a set in V
5 - any set including the depot
6 - any set not including the depot
The number of initial sets in strategies 1 to 4 is fixed. In the other strategies, we generate node sets randomly. In that case, the number of initial sets was fixed to 10 times the number of nodes.

The best results were obtained for strategy 6 and were improved by mixing it with strategy 1 (which is the fastest one). Computational results for the greedy shrinking algorithm can be found in Table 1.

We have also tested the possibility of, given a set S, adding to it all its neighbors, as well as selecting the node with higher demand, but the results were not conclusive.

Similar procedures to the above ones were developed independently by Araque et al. [AKMP94] for the identical customers CVRP.

Tabu search algorithm

Let S be a subset of V_0, we will denote by $N^+(S)$ the set of nodes in $V_0 \setminus S$ adjacent to at least one node in S and by $N^-(S)$ the set of nodes in S adjacent to at least one node in $V_0 \setminus S$. Given a set S for which d(S) and $x(\delta(S))$ have been computed in order to check violation in (2.4), note that it is easy to compute these quantities for every set $S \cup \{v\}, v \in N^+(S)$, and $S \setminus \{v\}$, for $v \in N^-(S)$.

At each iteration of the procedure a node set S is modified by removing or adding a node to it and (2.4) is checked. The procedure consists of two different phases, the expansion phase and the interchange phase, which are applied iteratively. At each iteration, in the expansion phase a node is added to the present set S, while in the interchange phase a node is added to (or removed from) S.

The expansion phase stops when no node can be added without violating $d(S) \leq (p + ulimit)C$, where ulimit is a given parameter, $0 < ulimit < 1$, and p takes values from 1 to $k-1$ all along the algorithm. In the interchange phase, the movements are selected in such a way that the resulting set S satisfies $(p - llimit)C \leq d(S) \leq (p + ulimit)C$, where llimit is another parameter, $0 < llimit < 1$.

Let us define $smax = C(p+ulimit) - d(S)$ and $smin = d(S) - C(p - llimit)$. Then, the set of candidate nodes to be added to S is $C^+(S) = \{v \in N^+(S) : d_v \leq smax\}$ while the set of candidates to be removed from S is $C^-(S) = \{v \in N^-(S) : d_v \leq smin\}$. If node v is finally added (removed) to (from) S, the reverse movement is declared tabu during a number of iterations equal to other input parameter called tll (tabu list length).

At any iteration in any phase, the aim is to select a movement in such a way that for the resulting set S, $x(\delta(S))$ be as small as possible. However, in the expansion phase, the node to be added to S is randomly selected among a set of "good" candidates (see step E.2 in the algorithm). Hence, by running the algorithm several times we may obtain different results. A parameter called ntimes determines the number of runs of the algorithm while another parameter, per, $0 \leq per < 1$, is used to define the set of "good" candidates.

The following algorithm is run starting with set $S = \{i\}$, for every node $i = 1, \ldots n$.

TABU ALGORITHM

Set $p = 1$.

Expansion phase

E.1. Compute $smax$ and $C^+(S)$. If $C^+(S)$ is empty, go to I.0 (Interchange phase).

E.2. Compute $M = max\{x((S:v)) : v \in C^+(S)\}$ and randomly select a node $v \in C^+(S)$ among those satisfying $M - per \leq x((S:v)) \leq M$.

E.3. Add v to S, check (2.4) and go to E.1.

Interchange phase

I.0. Set $iter = 1$.

I.1. Compute $smax, smin, C^+(S)$ and $C^-(S)$. Remove from $C^-(S)$ $(C^+(S))$ those nodes that have been added to (removed from) S in any of the last tll iterations. If $C^+(S) \cup C^-(S)$ is empty, go to I.4.

I.2. Let v be the node for which the following maximum is reached:

$$max\{\{x((S:j)), j \in C^+(S)\}, \{x((V_0 \setminus S:j)), j \in C^-(S)\}\}$$

I.3. Depending on if $v \in C^+(S)$ or $v \in C^-(S)$, add or remove it from S and check (2.4). Do $iter = iter + 1$. If $iter > tope$ go to I.4, otherwise, go to I.1.

I.4. Do $p = p + 1$. If $p \leq k - 1$ go to E.1, otherwise, stop.

The above procedure is run a number of times given by ntimes, that we have set between 1 and 3. After some testing, we have concluded that a good choice for the parameters is: $ulimit = 0.3, llimit = 0.1$, and $tll = 5$. If $ntimes \geq 2$, a good choice for per is 0.2, otherwise we set $per = 0$. Finally, the maximum number of iterations to be done in every interchange phase, tope, was chosen between 5 and 20.

The following tables compare the results obtained by the cutting plane algorithm when it uses different heuristics to identify violated capacity constraints. The heuristics we compare are: those explained at the beginning of this section which are due, mainly, to Harche and Rinaldi [HR91] (denoted HR in the tables) , the greedy shrinking algorithms and the tabu search with two different sets of values for the parameters; for tabu1, ntimes = 1 and tope = 10, while for tabu2, ntimes = 3 and tope = 20. Table 1 shows the lower bounds obtained using the different identification heuristics on a set of instances taken from the literature. Details of these instances and on the upper bounds shown, will be given in Section 6. Table 2 gives a statistical summary of these results. The term gap refers to the relative distance between the upper and lower bounds in percentage.

Name	Upper Bound	LB HR	LB shrinking	LB tabu1	LB tabu2
E-n101-k8	817	788.758	795.875	795.081	796.149
E-n22-k4	375	375	375	375	375
E-n23-k3	569	569	569	569	569
E-n30-k3	534	508.5	508.5	508.5	508.5
E-n33-k4	835	831.1	833.5	833.2	833.5
E-n51-k5	521	510.9	514.524	514.524	514.524
E-n76-k10	832	773.592	787.449	787.472	789.31
E-n76-k7	683	659.467	660.316	660.834	661.251
E-n76-k8	735	703.484	710.188	709.682	711.053
F-n135-k7	1165	1154.2	1155.89	1157.88	1157.55
F-n45-k4	724	723.8	723.667	724	724
F-n72-k4	238	232.5	232.5	232.5	232.5
M-n101-k10	820	818.165	819.412	819.333	819.333

Table 1

	LB HR	LB shrinking	LB tabu1	LB tabu2
average gap	2.22	1.86	1.85	1.81
maximum gap	4.8	4.8	4.8	4.8
number of wins	4	7	7	11
average number of cuts	339	946	1908	2299
average number iterations	50	33.5	29.7	31.2
average cpu time	171	182	700	859

Table 2

The best results were obtained by combining the greedy shrinking algorithm and tabu2. The strategy used was to call tabu2 only in those iterations where the greedy shrinking procedure fails to find a violated inequality. The results thus obtained are shown in Table 3.

Name	Upper Bound	Lower Bound	Gap	cpu lp solver	cpu identification	iterations
E-n101-k8	817	796.314	2.5	100.09	194.31	56
E-n22-k4	375	375	0	1.06	1.07	22
E-n23-k3	569	569	0	0.42	0.33	9
E-n30-k3	534	508.5	4.8	2.17	1.2	21
E-n33-k4	835	833.5	0.18	3.73	2.91	23
E-n51-k5	521	514.524	1.2	8.01	8.58	25
E-n76-k10	832	789.416	5.1	152.36	194.42	70
E-n76-k7	683	661.256	3.2	51.14	95.79	42
E-n76-k8	735	711.17	3.2	81.91	176.14	73
F-n135-k7	1165	1158.25	0.58	1198.07	314.79	123
F-n45-k4	724	724	0	5	2.78	27
F-n72-k4	238	232.5	2.3	8.56	2.87	19
M-n101-k10	820	819.5	0.061	138.1	42.26	46

Average gap: 1.78
Average total cpu time: 214
Average number of iterations: 42.8

Table 3

4 Separation procedures for other inequalities

4.1 Generalized capacity constraints.

We first describe procedures used to identify violation of inequality (2.9). The following procedure works on a shrunk graph, $HT(x)$, which initially is set equal to $GS(x)$, the graph obtained from $G(x)\backslash\{0\}$, after shrinking all the edges of value 1. At each iteration, an edge is shrunk and the current set of supernodes is used to define the partition Ω of V_0. The solution to the bin packing problem on Ω is denoted by $r(\Omega)$ and INCRE is a variable whose value is equal to the amount on which constraint (2.9) would be violated for Ω in the case $r(\Omega) = K+1$ would hold (it is very unlikely that $r(\Omega) > K+1$). If $INCRE \leq 0$, then $r(\Omega)$ need not be computed.

We denote by $d(u)$ to the demand of any node of the current shrunk graph, even if it is a supernode.

Procedure 4.1 *Global shrinking*

0) *Let $HT(x) = GS(x)$. Set $INCRE = 2$.*

1) *Let Ω be the partition defined by the set of nodes of $HT(x)$. If $INCRE > 0$, compute $r(\Omega)$; in the case $r(\Omega) > K$, constraint (2.9) is violated.*

2) *Select the edge of greatest value in HT(x). In the case that ties occur, we prefer edges that join supernodes or single nodes to supernodes. Let (u,v) be the edge selected and let y be its value.*

3) *Compute the new value of $INCRE$ as follows: $INCRE = INCRE + 2y - 2(\lceil \frac{d(u)}{C} \rceil + \lceil \frac{d(v)}{C} \rceil - \lceil \frac{d(u)+d(v)}{C} \rceil)$.*

4) *Shrink edge (u,v) and call $HT(x)$ the resulting graph. If $HT(x)$ has no edges, stop. Otherwise check the capacity constraint for the supernode just created and go to 1.*

The following procedure is very similar. It selects a node v and makes, initially, all the shrinkings on edges in its coboundary.

Procedure 4.2 *Local shrinking procedure*

In step 2 of the Global shrinking procedure, select the edge of greatest value in $HT(x)$ incident to supernode v until $INCRE \leq 0$ holds. At this moment, and until the end of the algorithm, apply the original step 2.

Now we present how to separate the bin inequality (2.10), that is the case where we do not use a partition of V_0 but only of a subset H. The following procedure is applied for each customer v.

Procedure 4.3 *(Bin inequality)*

0) *Choose randomly a set of $2k$ customers and add v to this set if v is not already in it.*

1) *Compute heuristically a maximum flow between this set of nodes and the depot. Let $(H : V \setminus H)$ be the resulting cut.*

2) *If $x(\delta(H)) < 2(\lceil \frac{d(H)}{C} \rceil + 1)$, apply the global shrinking procedure starting with $HT(x)$ as the graph induced by H in $GS(x)$ and $INCRE = 2(\lceil \frac{d(H)}{C} \rceil + 1) - x(\delta(H))$. Obviously, the violated inequalities found, if any, will be of type (2.10).*

Table 4 shows the lower bounds obtained with the cutting plane algorithm that uses the greedy shrinking and tabu2 algorithms for the identification of capacity constraints and all the above procedures to identify generalized capacity constraints. A group of procedures is called only when the previous ones have not found any violated inequality.

Name	Upper Bound	Lower Bound	Gap	cpu lp solver	cpu identification	iterations
E-n101-k8	817	796.314	2.5	120.69	263.71	56
E-n22-k4	375	375	0	1.3	1.5	22
E-n23-k3	569	569	0	0.63	0.5	9
E-n30-k3	534	532.5	0.28	6.18	8.83	50
E-n33-k4	835	833.5	0.18	4.36	4.08	23
E-n51-k5	521	514.524	1.2	9.56	12.57	25
E-n76-k10	832	789.416	5.1	175.98	254.56	70
E-n76-k7	683	661.256	3.2	62.39	130.27	42
E-n76-k8	735	711.17	3.2	99.6	232.48	73
F-n135-k7	1165	1158.25	0.58	1424.33	391.8	123
F-n45-k4	724	724	0	6.08	3.4	27
F-n72-k4	238	235	1.3	36.15	109.33	101
M-n101-k10	820	819.5	0.061	164.59	56.34	46

Average gap: 1.36
Average total cpu time: 275
Average number of iterations: 51.3

Table 4

The separation procedures for generalized capacity constraints are useful for only two instances (E-n30-k3, F-n72-k4). In both cases, the improvement is significant. It shows

that this class of inequality is important but also that its useness depends on the kind of instances. The instances E-n30-k3 and F-n72-k4 are special in the sense that some clients have very large demands. Thus the bin packing problem is crucial for these instances.

4.2 Combs.

The methods used to identify violated combs are similar to the ones used by Padberg and Rinaldi [PR90] and Grötschel and Holland [GH91] for the TSP. However, in the case of the CVRP, we consider also extended combs, in which the teeth and/or the handle can contain the depot. Furthermore, contrarily to the case of the TSP, the teeth have not to be necessarily disjoint sets. The identification has two stages; in the first one, we determine the handle, while in the second, the comb is completed by finding appropriate teeth and is checked for violation.

The following procedure is applied on a graph $G(x')$ obtained from $G(x)$ by removing some of its edges. For this purpose, several strategies have been applied:

a) Remove the edges with value less than or equal to eps or greater than or equal to 1-eps, for eps = 0, 0.1, 0.2 and 0.3.

b) In addition to a), remove all the edges incident with the depot.

c) In addition to a) or b), remove all the edges that join nodes which are inside sets corresponding to supernodes of the shrunk graph $GS(x)$.

d) In addition to a) or b), remove all the edges that join nodes which are inside sets, found heuristically or by other identification procedures, which satisfy:

- $x(\delta(V_0 \setminus S)) = 2r(V_0 \setminus S)$, if S contains the depot, or
- $x(\delta(S)) = 2r(S)$ and for some node $a \in S$: $r(S \setminus a) = r(S)$, if S does not include the depot.

Procedure 4.4 *(handle candidates)*

Find the biconnected components (blocks) of $G(x')$. The following sets are considered as handle candidates:

- *each block, and*
- *for each cut vertex v of $G(x')$, the union of blocks incident with v.*

For each handle, H, generated in the preceding procedure, we apply the following one to generate appropriate teeth.

Procedure 4.5 *(teeth candidates)*

1) *For every node $v \in H$, consider the following candidate teeth:*
   - *edges of G(x) incident with v and with the other endpoint not in H.*
   - *supernodes of graph $GS(x)$ containing v and not entirely in H if strategy c) has been applied to generate the handle H.*
   - *the sets of nodes for which all the edges among them have been removed in strategy d) to generate the handle H.*
   - *if node v is a cut vertex of G(x'), the following tentative teeth are also considered:*
     - *blocks incident with v and not in H,*
     - *semi-blocks not in H (graphs induced in G(x') by v and its neighbors in the blocks).*

   1.1) *Select the one with greatest value (the value of a teeth T is $x_T = x(\gamma(T)) + c(T) + 1 - |T|$, where c(T) has been defined in (2.13)).*

   1.2) *If its value is at least 0.5, the teeth is added to the comb, otherwise it is rejected. The best among all the rejected teeth is saved to be used in step 2).*

2) *If the number of added teeth is odd, go to 3). Otherwise, let T be the tooth with less value among those added to the comb, and let T' the tooth saved:*
   - *if $x_{T'} \geq 1 - x_T$, add the tooth T' to the comb.*
   - *if $x_{T'} < 1 - x_T$, or no tooth has been saved, removed the tooth T from the comb.*

3) *Check the resulting comb for condition (iii) and for violation.*

When a comb happens to be not violated, we try to enlarge its handle or its teeth such as it is described in Clochard and Naddef [CN94]

Results for this procedure are presented in Table 5. The lower bounds shown have been obtained using the greedy shrinking and tabu2 algorithms for the identification of capacity constraints and the above procedures to identify violated combs or extended combs (we do not identify the generalized capacity constraints). It should be noted that combs constraints have been useful in 7 instances, but improvements in the lower bounds are not too significant.

Name	Upper Bound	Lower Bound	Gap	cpu lp solver	cpu identification	iterations
E-n101-k8	817	798.358	2.3	203.18	741.33	96
E-n22-k4	375	375	0	1.37	3.56	22
E-n23-k3	569	569	0	0.61	2.7	9
E-n30-k3	534	508.5	4.8	2.71	3.96	21
E-n33-k4	835	833.5	0.18	4.41	6.28	23
E-n51-k5	521	517.111	0.75	14.75	89.92	42
E-n76-k10	832	790.792	5	245.88	547.34	102
E-n76-k7	683	662.661	3	85.48	400.54	82
E-n76-k8	735	712.415	3.1	133.98	494.2	107
F-n135-k7	1165	1158.48	0.56	1472.32	710.41	135
F-n45-k4	724	724	0	6.22	3.03	27
F-n72-k4	238	232.5	2.3	10.34	5.91	19
M-n101-k10	820	819.5	0.061	160.81	56.25	46

Average gap: 1.7
Average total cpu time: 416
Average number of iterations: 56.2

Table 5

4.3 Hypotours

We will present some heuristic algorithms to identify violated inequalities of this class. Let $E(x)$ be the edge set of the support graph $G(x)$, we describe now a two step algorithm for the basic hypotour inequality. In the first step, we check if $x(E \setminus E(x)) \geq 1$ is a valid inequality (and thus violated). We look for a node v such that the set of edges $E \setminus E(x)$ intersects every route containing v. If such node v is found, the second step of the algorithm strengthens the inequality by looking for a minimal set of edges F such that $F \subset E \setminus E(x)$ and $x(F) \geq 1$ is valid.

Note that if $E \setminus E(x)$ intersects every route including v, a path in $G(x)$ including v satisfies at least one of the two properties:

(i) the demand of the path is greater than C

(ii) at least one endpoint of the path is not adjacent to the depot in $G(x)$.

Our procedure is an implicit enumeration of all paths in $G(x)$ including v whose demand is lower than C. It is applied to every node.

Procedure 4.6 *(First step: Finding a violated inequality)*

Enumerate in $G(x)$ those paths including v whose demand is lower than the capacity C.

If a path whose endpoints are connected to the depot is found, stop. Otherwise, we say that an infeasibility has been detected.

If an infeasibility has been detected, the inequality $x(E \setminus E(x)) \geq 1$ is valid.

The paths are enumerated recursively; at each step we consider all the ways of enlarging a path in G(x) by adding two nodes. The number of possibilities is not too large because G(x) is in general rather sparse; furthermore, we limit the depth of the recursion by a given parameter T in order to avoid expensive computations.

Actually, the enumeration does not need to be complete. We compute first for $i = 1, \ldots, n$ the demand $md(i)$ of the minimum demand path in G(x) between the depot and node i. In the recursive algorithm, we do not need to continue enlarging a path $P = (a, \ldots, b)$ for which $d(V(P) \setminus \{a, b\}) + md(a) + md(b) > C$.

An easy modification of the procedure allows us to find other violated inequalities. Instead of stopping the procedure once a path connected to the depot is found, we allow to find a small number of such paths. At the end of the procedure, we look heuristically for an edge set F that intersects all these paths and such that $x(F) < 1$. If such a set is found, $x((E \setminus E(x)) \cup F) \geq 1$ is violated. In order to reduce computation time, only edges from $\delta(0)$ are considered to build F.

The following procedure tries to find a stronger inequality for those nodes v for which procedure (4.6) has found a violation. It computes a set of edges F including all edges in $E \setminus E(x)$ that allows to enlarge feasible paths in $G(x)$.

Procedure 4.7 *(Second step: Strengthening the inequality)*

Set $F = \emptyset$.

Apply procedure (4.6) to node v *with an extra treatment on each path* P *generated in the enumeration:*

Form the edge set $F_P = \{e \in (\delta(a_P) \cup \delta(b_P)) \setminus E(x) : d(V(P) \cup \{e\}) \leq C\}$ *(where* a_P *and* b_P *are the endpoints of* P*) and do* $F = F \cup F_P$.

If the enumeration in Procedure 4.7 is complete, the set of edges F given by this procedure intersects all feasible routes containing v and it is usually much smaller than $E \setminus E(x)$, so $x(F) \geq 1$ is a stronger inequality that is still violated.

Consider now the following extended hypotour inequalities. Let W be a set of customers, if H intersects all routes that induce a path on W (all customers in W are consecutive in the route), then the following inequality is valid:

$$x(\delta(W)) + 2x(H) \geq 4 \tag{4.16}$$

Let e (e and f) be an (two) edge(s) in $\delta(W)$ and let H intersecting all routes containing e (e and f) and inducing a path on W, then:

$$x(\delta(W)) + 2x(H) \geq 2x_e + 2 \tag{4.17}$$

and

$$x(\delta(W)) + 2x(H) \geq 2x_e + 2x_f \tag{4.18}$$

are valid inequalities, respectively.

Figure 4 shows a violated inequality of type (4.18) in which $W = \{6, 15, 17, 10, 14, 9, 18, 19\}$, edge $e = (6, 20)$, edge $f = (19, 4)$, the depot is denoted by node 1 and demands are in between parenthesis. Note that every route which induces a path on W and includes edges e and f, will have to connect nodes 20 and 4 to the depot and (due to the capacity), it could visit at most nodes 11 or 12 before getting to the depot. Then, it can be readily checked that every such a route will intersect the edge set $H = \{(1, 4), (12, 4), (1, 11)\}$.

Procedure 4.8 *(Extended procedure)*

Given a customer v, build greedily a node set W such that $x(\delta(W)) < 4$, $d(W) < C$ and $v \in W$.
Enumerate in $G(x)$ all paths covering the nodes of W. For each of these paths, call procedure 4.6 to enumerate all feasible extensions of it. If, for all of them, an infeasibility is detected, then call also procedure 4.7 to generate a set H such that the inequality (4.16) is violated.

Violated inequalities of types (4.17) and (4.18) are identified in a similar way.

Let $\Omega_1, \cdots, \Omega_p$, be a family of customer sets and let H be a set of edges that intersects all routes which induce a path on each of the sets $\Omega_1, \cdots, \Omega_p$, then the following inequality is valid.

$$\sum_{i=1,\cdots,p} x(\delta(\Omega_i)) + 2x(\delta(H)) \geq 2p + 2 \tag{4.19}$$

We identify violated inequalities of this type by applying procedures 4.6 and 4.7 on the shrunk graph $GS(x)$ at some stage of the application of procedure 4.1 (the first time INCRE is less than zero). The family of customer sets are those corresponding to supernodes of the shrunk graph.

Results for this procedure are presented in Table 6. The lower bounds shown have been obtained using the greedy shrinking and tabu2 algorithms for the identification of capacity constraints and the above procedures to identify hypotour inequalities.

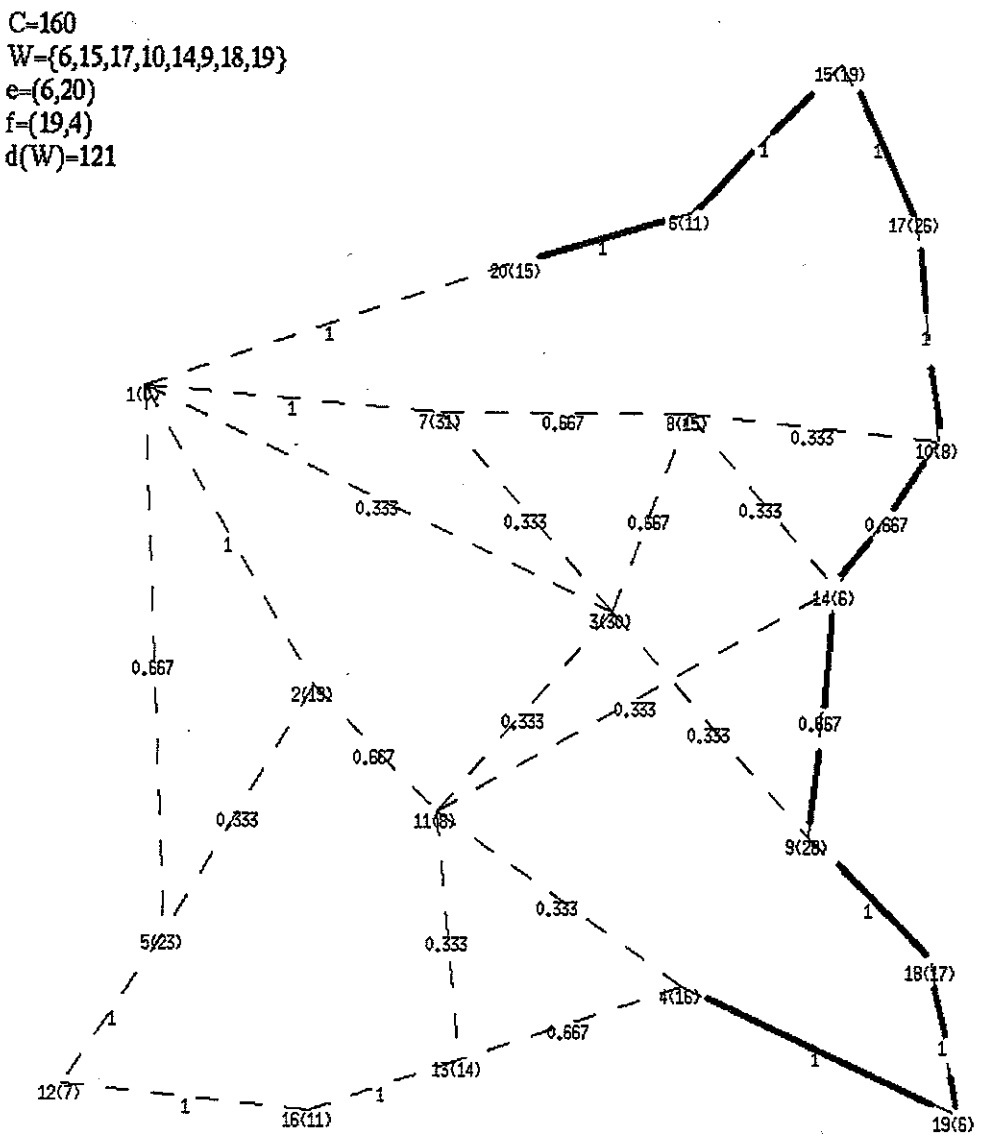

Figure 4: Violated hypotour inequality

Name	Upper Bound	Lower Bound	Gap	cpu lp solver	cpu identification	iterations
E-n101-k8	817	798.097	2.3	335.83	2355.99	199
E-n22-k4	375	375	0	1.36	1.54	22
E-n23-k3	569	569	0	0.58	0.44	9
E-n30-k3	534	509.833	4.5	3.48	3	26
E-n33-k4	835	834.154	0.1	7.03	30.81	42
E-n51-k5	521	515.96	0.97	26.38	152.09	77
E-n76-k10	832	793.089	4.7	588.8	1820.14	213
E-n76-k7	683	663.261	2.9	165.86	1112.45	158
E-n76-k8	735	712.578	3.1	210.34	1117.72	169
F-n135-k7	1165	1158.83	0.53	1631.58	857.36	147
F-n45-k4	724	724	0	5.79	3.2	27
F-n72-k4	238	233.6	1.8	14.75	15.27	33
M-n101-k10	820	820	0	167.99	83.71	54

Average gap: 1.61
Average total cpu time: 824
Average number of iterations: 90.5

Table 6

5 The Branch and Cut Algorithm

We have implemented a Branch and Cut algorithm for the CVRP, in the spirit of that by Padberg and Rinaldi [PR91] for the TSP.

The cutting plane algorithm described in the last sections is applied to every subproblem until no violated inequality is found or the solution did not increase during a certain number of iterations. Eventually, the subproblem will be fathomed if an integer feasible solution is found or the lower bound obtained is not less than the current upper bound. If the subproblem is not fathomed, it is divided in two subproblems by branching on certain inequality as it is explain below. The subproblem to be explored is then selected as in the LIFO strategy, that is, one of the two last subproblems just created is selected to be studied.

Branching strategies.

Padberg and Rinaldi [PR91] basic paper on branch and cut for the TSP, describes a branching strategy based on the selection of one edge variable with fractional value; then two subproblems will be created: one fixing that variable to zero and the other fixing it to one. The chosen variable is a variable whose value is as close as possible to 0.5 and whose coeffi-

cient in the objective function is as large as possible.

This branching strategy, called edge branching, appears to induce only local changes in the CVRP solutions and bounds. We tried also a more general strategy that we call branching on sets and that we expect to perturb the problem a little more. Let S be a set of nodes for which $x(\delta(S)) - 2r(S) = p(S), 0 < p(S) < 2$, then we can create two subproblems: one of them adding constraint $x(\delta(S)) = 2r(S)$ and the other adding $x(\delta(S)) \geq 2r(S) + 2$. Note that if $S = \{i, j\}$, this branching rule is equivalent to branching on edge $e = (i, j)$. Since we want to have a balanced branching tree, we consider as candidate sets, those sets for which $p(S)$ is between 0.75 and 1. These sets will be called odd sets.

The branching set selection is carried out in two steps: first, a candidate list of odds sets is build heuristically and, second, one of them is selected from that list according to some strategy. We have tested several strategies:

S1: Select the set S with maximum demand.

S2: Select the set S which contains the maximum number of supernodes in the shrunk graph GS(x).

S3: Select the set S which is farthest from the depot.

S4: Select the set S such that $x(\delta(S))$ is as close as possible from 3

S5: Select the set S such that $x(\delta(S))$ is as close as possible from 2.75

All these strategies happened to be better than the edge branching strategy. Yet, none of the five strategies overcomes the others.

A simple improvement to all these strategies is to use a linear programming selection scheme for selecting the candidate set. Applegate et al. [ABCC94] use successfully this method for the TSP. They first select a certain number of variables as close as possible from 0.5. For each variable, they solve the two corresponding subproblems and compute the minimum of the increases in the lower bounds with respect to that in the current node. Then, they choose the variable which leads to the maximum of these minimum increases. This can also be done in the case of the branching set selection. For instance in the case of strategy S1, we can select first the 6 sets with maximum demand and so on. Actually, the best improvement we obtain correspond to selecting one candidate set for each strategy S1 to S5 and using linear programming to select the final candidate. Denote by S6 this strategy and denote by S7 the strategy corresponding to the one of Applegate et al. with an initial set of ten edges. Finally, denote by S0 the usual edge branching strategy. Next table compares the results obtained when solving 15 instances both from the literature and randomly generated.

strategy	cpu time	# success	# nodes
S0	34h30	0	150
S5	14h55	0	76
S6	4h57	11	45
S7	14h	4	50

Table 7

For each strategy, the column "cpu time" gives the total cpu time. The column "# success" gives the number of times the strategy is the best. The column "# nodes" gives the mean number of nodes in the branching tree.

It seems obvious that strategy S6 have to be used for the CVRP. It shows the interest of a linear programming selection scheme but also the interest of using sets in this scheme. It is not possible to show the same results concerning the hardest instances in the litterature since we can not solve them to optimality. Yet, we have computed the number of nodes and time needed to decrease the gap between the lower and upper bound by 0.5%. Compared to strategy S0 and applied to 20 unsolved instances, strategy S6 allows to divide the number of nodes by 21 and the cpu time by 17 while strategy S5 allows to divide the number of nodes by 3.5 and the cpu time by 5.

6 Computational Results.

The procedures described in this paper have been applied to a set of 13 difficult instances taken from the literature. Table 8 shows for each test instance: number of cities (including the depot), number of vehicles and the tightness of the capacity constraints (total demand of the customers divided by the total capacity of the fleet of vehicles). Also, it shows the reference from which each instance have been taken and where the complete data can be found. Data for the first nine instances can be also found in the TSPLIB (Reinelt [Rei91]).

Name	cities	vehicles	Tightness	Reference
E-n101-k8	101	8	0.91	[CE69]
E-n22-k4	22	4	0.94	[CE69]
E-n23-k3	23	3	0.75	[CE69]
E-n30-k3	30	3	0.94	[CE69]
E-n33-k4	33	4	0.92	[CE69]
E-n51-k5	51	5	0.97	[CE69]
E-n76-k10	76	10	0.97	[CE69]
E-n76-k7	76	7	0.89	[CE69]
E-n76-k8	76	8	0.95	[CE69]
F-n135-k7	135	7	0.95	[Fis94]
F-n45-k4	45	4	0.90	[Fis94]
F-n72-k4	72	4	0.96	[Fis94]
M-n101-k10	101	10	0.91	[CMT79]

Table 8

All the test instances are planar, that is, customers are located at points in the plane and $b_{ij} = \lfloor c_{ij} + \frac{1}{2} \rfloor$, where c_{ij} is the euclidean distance between points i and j. This is the same cost function as the one proposed in the TSPLIB. Other authors, as Fisher [Fis94], have prefered to use real costs, so it is difficult to compare the results obtained by different methods even if the test instances are the same.

Table 9 presents the lower bounds obtained by our cutting plane algorithm on the above set of instances, as well as the cpu times, in seconds, in a Sun Sparc 10 machine. The identification procedures used were the greedy shrinking and tabu2 for the capacity constraints and all the procedures presented here for the identification of generalized capacity constraints, combs and hypotours. The upper bounds in the table were provided to us by Campos and Mota [CM95] and were computed with two heuristic algorithms based on tabu search and on the use of LP solutions. Also shown are the gap between the upper and the lower bounds and, for comparison purposes, the gap and cpu times obtained by Fisher on some of these instances. Note that the costs used by Fisher are real ones and therefore the lower bounds should not be compared directly because the optimal costs may not be the same; nevertheless, the comparison between the respective gaps may override, at least partially, this difficulty. In a recent paper, Mingozzi et al. [MCH94] reports results on three instances of this set; gaps obtained were: 0 for E-n22-k4, 0.71 for E-n51-k5 and 2.24 for E-n76-k10 that was the largest instance tried in that paper.

Name	Upper Bound	Lower Bound	Gap	cpu [a]	Gap in [Fis94]	cpu [b] in [Fis94]
E-n101-k8	817	799.656	2.1	1708	4.87	18477
E-n22-k4	375	375	0	5		
E-n23-k3	569	569	0	4		
E-n30-k3	534	534	0	30		
E-n33-k4	835	835	0	46		
E-n51-k5	521	517.142	0.74	129	3.34	5745
E-n76-k10	832	793.384	4.6	1919	9.55	11038
E-n76-k7	683	664.355	2.7	1052		
E-n76-k8	735	713.746	2.9	1282		
F-n135-k7	1165	1159.06	0.51	2024	2.57	15230
F-n45-k4	724	724	0	12	0.38	2984
F-n72-k4	238	235	1.26	59	1.74	6301
M-n101-k10	820	820	0	167	0.22	15578

(a) seconds in a Sun Sparc 10
(b) seconds in a Apollo Domain 3000

Table 9

Six instances are solved using the cutting plane algorithm. We manage to solve three more using the Branch and Cut algorithm. These results are presented in Table 10. Note that the F-n135-k7 instance had never been solved to optimality before and is the largest instance ever solved in the literature.

Name	Optimal value	cpu time (a)	# nodes (b)
E-n51-k5	521	342.1	9
F-n135-k7	1162	18871	95
F-n72-k4	237	319.8	57

(a) seconds in a Sun Sparc 10
(b) number of nodes of the enumeration tree

Table 10

Acknowledgments: We thank V. Campos, J. M. Clochard, M. C. Martinez, E. Mota, Y. Pochet, J. M. Sanchis and L. Wolsey for a lot of comments that have greatly benefited the content of this paper.

References

[ABCC94] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. Special session on TSP. In *15th International Symposium on Mathematical Programming.* University of Michigan, USA, Aug 1994.

[AHM90] J. R. Araque, L. Hall, and T. Magnanti. Capacitated trees, capacitated routing and associated polyhedra. Discussion paper 9061, CORE, Louvain La Neuve, 1990.

[AKMP94] J.R. Araque, G. Kudva, T.L. Morin, and J.F. Pekny. A branch-and-cut for vehicle routing problems. *Annals of Operations Research*, 50:37–59, 1994.

[AMS89] Y. Agarwal, K. Mathur, and H. M. Salkin. Set partitioning approach to vehicle routing. *Networks*, 7:731–749, 1989.

[AP95] P. Augerat and Y. Pochet. New valid inequalities for the vehicle routing problem. In preparation, 1995.

[Ara90] J. R. Araque. Solution of a 48-city vehicle routing problem by branch and cut. Research Memorandum 90-19, Purdue University, 1990.

[BGAB83] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and Scheduling of vehicles and crews: the state of the art. *Computers and Operations Research*, 10:69–211, 1983.

[CCM91] V. Campos, A. Corberán, and E. Mota. Polyhedral results for a vehicle routing problem. *European Journal of Operational Research*, 52:75–85, 1991.

[CE69] N. Christofides and S. Eilon. An algorithm for the vehicle dispatching problem. *Operations Research Quarterly*, 20:309–318, 1969.

[CH93] G. Cornuejols and F. Harche. Polyhedral study of the capacitated vehicle routing. *Mathematical Programming*, 60:21–52, 1993.

[Chr85] N. Christofides. Vehicle Routing. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 431–448. John Wiley & Sons Ltd., Chichester, 1985.

[CM95] V. Campos and E. Mota. Obtaining feasible solutions for the Vehicle Routing Problem in a Branch and Cut scheme. In preparation, 1995.

[CMT79] N. Christofides, A. Mingozzi, and P. Toth. "The Vehicle Routing Problem". In N. Christofides, A. Mingozzi, P Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 318–338. John Wiley & Sons Ltd., Chichester, 1979.

[CN94] J.-M. Clochard and D. Naddef. Some fast and efficient heuristics for comb separation in the symetric traveling salesman problem. Technical Report RR941M, submitted to Mathematical Programming, Institut IMAG, Grenoble, 1994.

[Fis94] M. Fisher. Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees. *Operations Research*, 42(4):626–642, 1994.

[GH91] M. Grötschel and O. Holland. Solution of large-scale symmetric travelling salesman problems. *Mathematical Programming*, 51:141–202, 1991.

[GHL94] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 1994.

[GP79] M. Grötschel and M.W. Padberg. On the Symmetric Traveling Salesman Problem: I and II. *Mathematical Programming*, 16:265–280 and 281–302, 1979.

[GP85] M. Grötschel and M. W. Padberg. Polyhedral Theory. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 251–305. John Wiley & Sons Ltd., Chichester, 1985.

[HASG94] D.T. Hiquebrau, A.S. Alfa, J.A. Shapiro, and D.T. Gittoes. A Revised Simulated Annealing and Cluster-First Route-Second Algorithm Applied to the Vehicle Routing Problem. *Engineering Optimization*, 22:77–107, 1994.

[HR91] F. Harche and G. Rinaldi. Vehicle Routing. Private communication, 1991.

[Lap92] G. Laporte. The Vehicle Routing Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

[LN84] G. Laporte and Y. Nobert. Comb Inequalities for the Vehicle Routing Problem. *Methods of Operations Research*, 51:271–276, 1984.

[LND85] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1058–1073, 1985.

[Mag81] T.L. Magnanti. Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects. *Networks*, 11:179–213, 1981.

[MCH94] A. Mingozzi, N. Christofides, and E. Hadjiconstantinou. An exact algorithm for the vehicle routing problem based on the set partitionning formulation. June 1994.

[Osm93a] I. H. Osman. Vehicle Routing and Scheduling: Applications, algorithms and developments. Technical report, Institute of Mathematics and Statistics, University of Canterbury, 1993.

[Osm93b] I.H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

[PR90] M. W. Padberg and G. Rinaldi. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47:219–257, 1990.

[PR91] M. W. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[Rei91] G. Reinelt. TSPLIB: A traveling salesman problem library. *ORSA Journal of Computing*, 3:376–384, 1991.

[Tai93] É. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–674, 1993.